Proceedings of the ASME 2022 17th International Manufacturing Science and Engineering Conference
MSEC2022
June 27-July 1, 2022, West Lafayette, Indiana, USA

MSEC2022-85269

COMPARISON OF HUMAN SKELETON TRACKERS PAIRED WITH A NOVEL SKELETON FUSION ALGORITHM

Jared T. Flowers
University of Florida
Gainesville, FL

Gloria J. Wiens
University of Florida
Gainesville, FL

ABSTRACT

The onset of Industry 4.0 brings a greater demand for Human-Robot Collaboration (HRC) in manufacturing. This has led to a critical need for bridging the sensing and AI with the mechanical-n-physical necessities to successfully augment the robot's awareness and intelligence. In a HRC work cell, options for sensors to detect human joint locations vary greatly in complexity, usability, and cost. In this paper, the use of depth cameras is explored, since they are a relatively low-cost option that does not require users to wear extra sensing hardware. Herein, the Google Media Pipe (BlazePose) and OpenPose skeleton tracking software packages are used to estimate the pixel coordinates of each human joint in images from depth cameras. The depth at each pixel is then used with the joint pixel coordinates to generate the 3D joint locations of the skeleton. In comparing these skeleton trackers, this paper also presents a novel method of combining the skeleton that the trackers generate from each camera's data utilizing a quaternion/link-length representation of the skeleton. Results show that the overall mean and standard deviation in position error between the fused skeleton and target locations was lower compared to the skeletons resulting directly from each camera's data.

Keywords: smart manufacturing systems, skeleton tracking, human-robot interaction, cobot, control and automation.

NOMENCLATURE

${}^{k}P_{m_i}$	Point in cartesian space, $m \in \{h, r\}$ for point on the human or robot, $i \in \{1,2,3\}$ for point number, $k \in \{A, B\}$ for the camera frame (A or B) in which the point is defined.
$\boldsymbol{{}^{k}x_n}$, $\boldsymbol{{}^{k}y_n}$, $\boldsymbol{{}^{k}z_n}$	The x, y, or z axis of coordinate frame $n \in \{H, R\}$ (human or robot) as seen in coordinate frame $k \in \{A, B\}$ (camera A or B). A tilde over the axis vector indicates it has been unitized.
${}^{k}_{n}R$	The 3x3 rotation matrix that relates frame $n \in \{H, R\}$ (human or robot) to frame $k \in \{A, B\}$ (camera A or B).
${}^{k}_{n}T$	The 4x4 homogenous transformation matrix that relates frame $n \in \{H, R\}$ (human or robot) to frame $k \in \{A, B\}$ (camera A or B).
q_l	Quaternion for the l^{th} link of the human kinematic chain.
$\boldsymbol{m_l}$	Unit vector that is the axis of rotation for q_l.
$\boldsymbol{v_l}$	Vector for the l^{th} link of the human kinematic chain.
P_{distal}	The joint farthest from the torso for a particular link in the human kinematic chain.
$P_{proximal}$	The joint closest to the torso for a particular link in the human kinematic chain.
$\boldsymbol{ll_{fused}}$	Column vector of link lengths of the fused skeleton.
$\boldsymbol{ll_{camera}}$	Column vector of link lengths for the skeleton from $camera \in \{A, B\}$ (camera A or B).
$\hat{\boldsymbol{x}}_{\boldsymbol{n,n}}$, $\dot{\hat{\boldsymbol{x}}}_{\boldsymbol{n,n}}$, $\ddot{\hat{\boldsymbol{x}}}_{\boldsymbol{n,n}}$	Kalman filter estimated quaternion elements, derivative of estimates, and second derivative of estimates, respectively.
θ_l	Rotation angle about $\boldsymbol{m_l}$ defined by quaternion q_l.
$\alpha, \beta, \gamma, \delta$	Kalman filter gains controlling rate of change of $\hat{\boldsymbol{x}}_{\boldsymbol{n,n}}$, rate of change of $\dot{\hat{\boldsymbol{x}}}_{\boldsymbol{n,n}}$, rate of change of $\ddot{\hat{\boldsymbol{x}}}_{\boldsymbol{n,n}}$, and decay rate of $\dot{\hat{\boldsymbol{x}}}_{\boldsymbol{n,n}}$ and $\ddot{\hat{\boldsymbol{x}}}_{\boldsymbol{n,n}}$, respectively.

1. INTRODUCTION

Since the introduction of Industry 4.0, human-robot collaboration (HRC) has been promoted as an enhancement to manufacturing work cells that can allow humans and robots to work together synergistically to complete tasks that require the unique capabilities of humans and robots. Some HRC tasks require tight coordination between the robot and human, such as a human-robot part handover, and therefore require very precise

sensing of human hand locations. Other HRC tasks may not involve as much coordination between robot and human, such as a robot pick-and-place task, but still require detection of human locations to ensure human safety among industrial robots. Tasks not requiring robot-human coordination could utilize a method of detection with more coarse resolution compared to tightly coordinated tasks. Moreover, high validity in detection of human locations will allow for proactive and reactive robot responses to human motion, allowing robots to avoid collisions with humans while completing interactions such as human-robot handovers.

A recent trend in HRC is to use predicted data to either determine immediate robot motion or to determine if changes in the robot sequence are necessary due to predicted human motion. Many works in this area depend on receiving human joint locations as input. In the algorithm presented in [1], the estimated handover location, which is the output of a recurrent neural network (RNN), will only be reliable if the real-time human joint locations used as input are accurate. In [2], motion onset detection and human intent estimation is attained by using probabilistic principal component analysis and probabilistic motion primitives methods on human joint angle trajectories as input. This requires the human joint locations to be estimated first. The work in [3] presents another application of neural networks to human intent prediction using human gaze and pose data as input. The pose part of the input requires accurate human joint locations. The algorithm presented in [4] also requires human joint locations as input to a system of convolutional neural networks that classify actions a human is performing in front of an RGB-D camera. The Risk-of-Passage algorithm developed in [5] takes human joint location as input and estimates the risk a robot will incur by proceeding with passage between human links. The robot motion segmentation framework developed in [6] also utilizes human joint locations as an input and determines proactive-n-reactive robot behaviors upon human interruption.

State-of-the art solutions for detecting human joint locations use sensors such as laser scanners or motion-capture systems which are relatively expensive compared to depth cameras. Additionally, some sensor suites (e.g., motion-capture systems) require the worker in a HRC work cell to wear a suit with many tracking markers [7]. In [8], another sensor suite is proposed in which the human wears several inertial measurement units ranging from 8 to 17 and several RGB cameras ranging from 2 to 8. Sensor suites requiring a worker to wear many accessories can be cumbersome, impractical and may even restrict worker motion and/or require significant setup time. Therefore, a relatively inexpensive solution for human tracking that does not require any extra setup before a worker enters the work cell is desirable for HRC.

A few skeleton tracking software packages exist that can determine human joint locations given RGB images from a camera as input. These software packages can determine the pixel coordinates of human joints, which can then be converted to 3D coordinates using the depth channel from a depth camera. Some of these methods can also estimate 3D coordinates directly from the RGB image, but still require the distance between the camera and at least one point on the human to relate those 3D coordinates to the world reference frame. Google Media Pipe Pose Tracking (BlazePose) [9] and OpenPose [10] are representative of different existing skeleton tracking methods and therefore are selected for comparison in this paper. These skeleton trackers utilize deep neural networks that take RGB image data as input and output the pixel coordinates of each human joint, as well as keypoints on the face. As with any vision system, even when using advanced skeleton tracking algorithms, a single camera cannot provide accurate or complete results when portions of a human become occluded by objects within the work cell and/or by the robot. Noise in the depth camera's sensed images can also prevent skeleton tracking software from accurately determining the locations of human skeleton joints. Therefore, using multiple cameras provides a necessary level of redundancy in skeleton tracking, where fusion of the joint location data can more reliably yield complete human skeletons for tracking human movement within the work cell.

A few works have focused on the area of skeleton tracking and fusion with multiple depth cameras as input. Takahashi et al. [11] proposed an approach which uses data from 2 RGB cameras to generate 3D human poses. They minimized error in rotation, translation, human link lengths, and human motion by optimizing temporal difference parameters. Experiments showed a 5.4m error in determining the position of cameras relative to each other, on a 100m long field. Chen et al. [12] proposed a method of skeleton fusion which uses human joint locations from 2 Kinect sensors as input. Their method first determines the validity of joint locations based on how close human link lengths are to averages taken over 100 frames. If a joint location is invalid, then its location is inferred based on valid non-adjacent joints in the kinematic chain and the human link lengths. Finally, the coordinates for each joint of the fused skeleton are simply taken from the camera that produced highest joint validity, per each joint. Experimental results in [12] show the transform relating camera positions achieved accuracy up to 4.1mm.

In [13], Moon et al. also developed a multiple Kinect camera skeleton tracking system that uses a Kalman filter to determine the validity of each joint location in each camera based on the level of noise in the joint location estimation. Again, the fused joint location for each joint was taken from the camera data producing the highest validity for that joint. Their experiments utilized 5 Kinect cameras and achieved average joint position error of 6.95cm. In [14], Huang et al. used OpenPose to extract 2D human joint coordinates from many Kinect cameras. Their method of skeleton fusion uses a ray tracing approach to determine if a human joint is occluded in a camera's frame and estimate the actual location of the occluded joint using a weighted average of the joint location from all cameras. They also utilized a Kalman Filter like approach with prior distributions on the estimation of a joint's location based on the location of the previous joint in the human kinematic chain. Their experiments achieved a mean joint localization error of 5.68cm. The method presented, herein this paper, also makes use of a version of the Kalman filter, which has been used for state estimation in random environments [15,16]. In the Kalman filter,

state observations are computed based on previous state observations and then state estimates are computed based on probability and previously observed state values.

In comparing skeleton trackers, this paper presents a method of fusing human skeletons generated from 2 depth cameras; improving tracking accuracy and validity. The proposed method differs from previously developed methods by decomposing the human into a quaternion and link length representation. Then the fused skeleton is reconstructed using the best quaternion from any camera for each link of the fused skeleton. The quaternions of the fused skeleton are then passed through a low pass Butterworth filter to reduce noise and then a Kalman filter for state estimation before using forward kinematics to generate the fused skeleton joint locations. This work tests three methods of skeleton tracking and compares those methods based on the accuracy of joint locations directly from each camera's data, without affects from the skeleton fusion. The skeleton fusion method is also applied to the trackers' output, and accuracy of the fused joint location estimations is compared to the joint locations directly from each camera's data when using two cameras for input with occlusions occurring due to the robot manipulator.

The remainder of the paper is structured as follows: section 2.1 introduces each skeleton tracking software package used, 2.2 presents a camera calibration method, 2.3 presents the method of decomposing a human into a quaternion representation, 2.4 presents the method of fusing the skeletons from each camera, 2.5 presents the filtering method, and 2.6 explains skeleton quaternion/link length forward kinematics. Section 3 provides experimental results and a comparative discussion highlighting potential sources of error followed by a conclusion in section 4.

2. METHODS

The flow of data of the proposed method for generating human skeleton data is shown in Fig. 1. It shows the depth camera outputs an RGB image which is passed to the deep convolutional neural network (DCNN) in each skeleton tracker which then outputs the human's joint location in pixel coordinates. Then the depth image output from the depth camera is used to convert the 2D pixel coordinates into 3D cartesian coordinates. Next, nominal link lengths and quaternions for the human skeleton are determined by each camera. Then, from among all the cameras' link lengths and quaternions, the skeleton fusion algorithm selects the 'best' quaternions to represent each link, stitching (fusing) them together to generate the skeleton. Finally, these stitched (fused) together quaternions are passed through filtering algorithms, resulting in the fused skeleton.

2.1 Skeleton Tracking

For the proposed method of fusing human skeleton models from data of multiple depth sensors, open-source skeleton tracking software was used to provide human joint locations as output, given RGB+Depth images as input. Herein, this paper, these input images were captured from 2 depth cameras. Google Media Pipe Pose Tracking (Media Pipe) and OpenPose were the skeleton tracking software packages tested. Both utilize DCNNs,

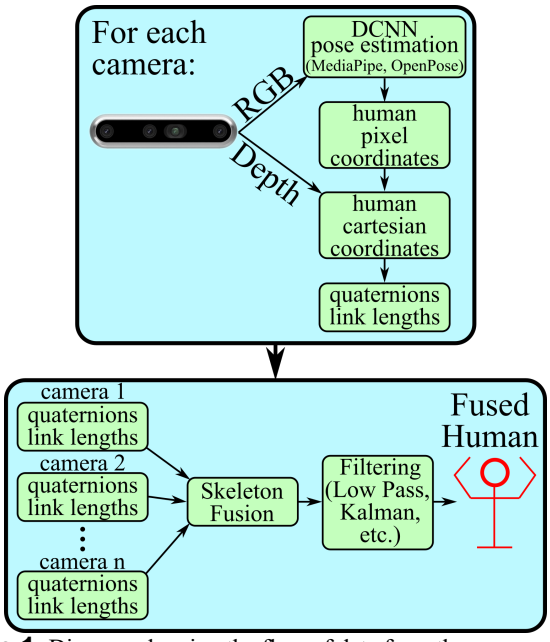

Figure 1. Diagram showing the flow of data from the camera output to the fused human skeleton model.

each with different architectures, to output human joint pixel coordinates when provided RGB images as input. Media Pipe is a top-down network, detecting regions of the image that might contain a human and then detecting joint locations within each region [17]. OpenPose is a bottom-up network, detecting all joint locations and then determining how to group the joints for each human. Media Pipe is built upon Google's BlazePose network and has been found to have a Percentage of Correct Keypoints (PCK) score of up to 97.5% keypoints within 0.2 torso diameters (PCK0.2) of the correct location while performing a HIIT task [18]. Results in [9] showed on average OpenPose having a PCK0.2 score that was 3.7% higher than BlazePose, but BlazePose computed about 78 times faster than OpenPose. The skeleton trackers also differ in the number of people that can be tracked: 1 person with Media Pipe, and unlimited (with no computer hardware constraints) with OpenPose. The neural network in the Media Pipe algorithm also has the capability to estimate 3D coordinates of human joints from RGB image input, with the midpoint of the hips being the origin of the Media Pipe coordinate frame. The depth channel from the cameras can be used to locate the midpoint of the hips in the world coordinate frame, allowing the Media Pipe 3D joint location estimates to be easily transformed to the world coordinate frame. Therefore, Media Pipe with the 3D joint location estimation is a third method tested in this paper.

2.2 Camera Calibration

For the coordinates determined from the skeleton trackers to be useful, the relationship between each camera and the world coordinate frame must be determined. In a multiple camera system, if the skeleton of the same person is detected from each camera's frame, then at least 3 points (joint locations) that are not colinear are common between camera frames. Therefore, the

homogenous transformations that relate the position of the cameras to each other can be determined. To determine these transformations, consider 2 cameras, A and B, where the homogenous transformation ${}^{A}_{B}T$ transforms points from camera B's coordinate frame to camera A's coordinate frame. Additionally, the 3 common points in each camera frame will be denoted P_{h_1}, P_{h_2}, and P_{h_3}, as shown in Fig. 2. A new coordinate frame denoted H will be determined using these common points. Figure 2 shows the 3 common points on the human as well as the axes of frame H. A homogenous transformation between each camera (k) and frame H becomes:

$$\boldsymbol{{}^{k}x_H} = {}^{k}P_{h_2} - {}^{k}P_{h_1} \quad (1)$$

$$\boldsymbol{{}^{k}z_H} = \boldsymbol{{}^{k}x_H} \times \left[{}^{k}P_{h_3} - {}^{k}P_{h_1}\right] \quad (2)$$

$$\boldsymbol{{}^{k}y_H} = \boldsymbol{{}^{k}z_H} \times \boldsymbol{{}^{k}x_H} \quad (3)$$

$${}^{k}_{H}R = \left[\boldsymbol{{}^{k}\tilde{x}_H}, \boldsymbol{{}^{k}\tilde{y}_H}, \boldsymbol{{}^{k}\tilde{z}_H}\right],\ {}^{k}_{H}T = \begin{bmatrix} {}^{k}_{H}R, {}^{k}P_{h_1} \\ 0,0,0,1 \end{bmatrix} \ for\ k = \{A, B\} \quad (4)$$

Now, ${}^{A}_{B}T = {}^{A}_{H}T\,{}^{H}_{B}T$ where ${}^{H}_{B}T = \left[{}^{B}_{H}R^{T}, -{}^{B}_{H}R^{T\,B}P_{h_1}; 0,0,0,1\right]$. Now that the transformation between any 2 cameras can be determined, the transformations relating each camera's coordinate frame to that of a main camera can be determined. To calibrate the two cameras used for testing to each other, ${}^{A}_{B}T$ was determined for each of 500 frames. Then the set of all ${}^{A}_{B}T$ transformations was averaged. In determining ${}^{A}_{B}T$, the pelvis and each wrist were used for P_{h_1}, P_{h_2}, and P_{h_3}, respectively. Considering only the torso, these points are the combination of 3 points that maximizes the distance between points and generated the most precise average ${}^{A}_{B}T$.

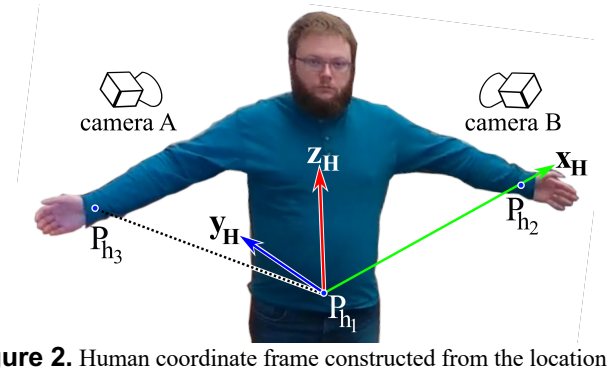

Figure 2. Human coordinate frame constructed from the location of the pelvis and wrists.

In addition to the camera-camera relationships, the transformation from the main camera to the world coordinate frame must be determined. Industrial robot manipulators can be configured such that 3 points on the robot are non-colinear, denoted P_{r_1}, P_{r_2}, and P_{r_3}, illustrated in Fig. 3. Additionally, if the manipulator can provide feedback of the variable joint parameters, then forward kinematics can be used to determine the location of those 3 non-colinear points on the robot in the world coordinate frame. If the robot points can be detected in the main camera's frame, then a transformation between the main camera and a new frame, denoted frame R, generated by the robot points, can be determined. Figure 3 shows 3 points on a robot manipulator used to generate frame R, which is also shown. The world (fixed) coordinate frame, frame F, is also shown at the base of the robot. The transformation between the world coordinate frame or the main camera's frame and the robot points, denoted ${}^{F}_{R}T$ and ${}^{A}_{R}T$, can be determined by:

$$\boldsymbol{{}^{k}x_R} = {}^{k}P_{r_2} - {}^{k}P_{r_1} \quad (5)$$

$$\boldsymbol{{}^{k}z_R} = \boldsymbol{{}^{k}x_R} \times \left[{}^{k}P_{r_3} - {}^{k}P_{r_1}\right] \quad (6)$$

$$\boldsymbol{{}^{k}y_R} = \boldsymbol{{}^{k}z_R} \times \boldsymbol{{}^{k}x_R} \quad (7)$$

$${}^{k}_{R}R = \left[\boldsymbol{{}^{k}\tilde{x}_R}, \boldsymbol{{}^{k}\tilde{y}_R}, \boldsymbol{{}^{k}\tilde{z}_R}\right],\ {}^{k}_{R}T = \begin{bmatrix} {}^{k}_{R}R, {}^{k}P_{r_1} \\ 0,0,0,1 \end{bmatrix} \ for\ k = \{F, A\} \quad (8)$$

Now ${}^{F}_{A}T = {}^{F}_{R}T\,{}^{R}_{A}T$ where ${}^{R}_{A}T = \left[{}^{A}_{R}R^{T}, -{}^{A}_{R}R^{T\,A}P_{r_1}; 0,0,0,1\right]$. After computing ${}^{F}_{A}T$ and ${}^{A}_{B}T$, points in the second camera (camera B) can now be transformed to the fixed frame by ${}^{F}_{B}T = {}^{F}_{A}T\,{}^{A}_{B}T$. Now, human joint locations can be transformed from each camera's frame to the world coordinate frame before computing the human quaternion representation for each camera using:

$${}^{F}P_{h_i} = {}^{F}_{k}T\,{}^{k}P_{h_i}\ for\ k = \{A, B\} \quad (9)$$

for the i^{th} joint of the human in the frame of camera A or B.

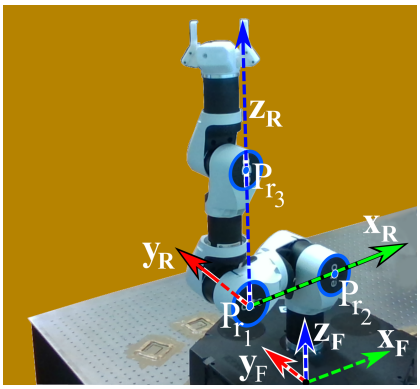

Figure 3. Three points used to determine the transformation from the main camera to the world coordinate frame.

2.3 Skeleton Quaternion/Link-Length Representation

The proposed method determines the quaternions and link lengths that relate each human link to the fixed coordinate frame. This contrasts with other methods that use joint cartesian coordinates directly, which may not maintain reasonable skeleton dimensions. Quaternions provide an axis-angle representation of the rotation that relates one sliding vector to another sliding vector. Quaternions are defined by:

$$\boldsymbol{q_l} = \langle w_l, x_{q_l}, y_{q_l}, z_{q_l} \rangle \quad (10)$$

for the l^{th} link of the human kinematic chain. The human kinematic chain is shown in Fig. 4 and described in Table 1, showing link indices and proximal/distal joint for each link. The w_l, x_{q_l}, y_{q_l}, z_{q_l} are common variable names for parts of the quaternion, not cartesian coordinates. The parts of the quaternion can be expressed by:

$$w_l = \cos\left(\frac{\theta_l}{2}\right) \text{ and } \langle x_{q_l}, y_{q_l}, z_{q_l} \rangle = \sin\left(\frac{\theta_l}{2}\right) \boldsymbol{m_l} \quad (11)$$

where θ_l is the angle between the link and the world z-axis and $\boldsymbol{m_l}$ is the unit vector about which rotation occurs, represented by:

$$\boldsymbol{m_l} = \frac{x_{q_l}\hat{\imath}+y_{q_l}\hat{\jmath}+z_{q_l}\hat{k}}{\sqrt{x_{q_l}^2+y_{q_l}^2+z_{q_l}^2}}. \quad (12)$$

To determine the rotation angles and axis between human links, the human kinematic chain is iterated over according to:

$$\boldsymbol{v_l} = P_{distal} - P_{proximal}, \text{ and } \boldsymbol{z_f} = [0,0,1]^T \quad (13)$$

$$\boldsymbol{m_l} = \boldsymbol{v_l} \times \boldsymbol{z_f} \rightarrow \boldsymbol{m_l} = \frac{\boldsymbol{m_l}}{\|\boldsymbol{m_l}\|} \quad (14)$$

where subscript l denotes a link of the human kinematic chain. The P_{distal} is the human joint farthest from the torso in the kinematic chain and $P_{proximal}$ is the human joint closest to the torso in the kinematic chain, for each link. The latter part of Eq. 14 is indicating $\boldsymbol{m_l}$ is normalized. The rotation angle between the human link and the world z-axis about $\boldsymbol{m_l}$ is computed by:

$$\theta_l = atan2\left(\frac{(\boldsymbol{v_l}\times\boldsymbol{z_f})}{\|\boldsymbol{v_l}\|}, \frac{(\boldsymbol{v_l}\cdot\boldsymbol{z_f})}{\|\boldsymbol{v_l}\|}\right). \quad (15)$$

Table 1. Links of the human kinematic chain.

Link	Description	Proximal	Distal
1	Spine	Pelvis (Hip MP)	Spine
2	Neck	Spine	Shoulder MP
3	Shoulder to Shoulder	Shoulder MP	Shoulders
4	Left Upper Arm	Left Shoulder	Left Elbow
5	Left Forearm	Left Elbow	Left Wrist
6	Right Upper Arm	Right Shoulder	Right Elbow
7	Right Forearm	Right Elbow	Right Wrist

MP=midpoint

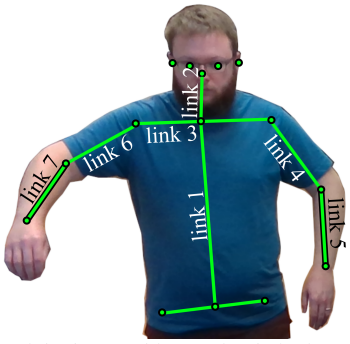

Figure 4. Links of the human kinematic chain from the waist up.

Figure 5 shows the rotation vector, $\boldsymbol{m_7}$, and rotation angle, θ_7, between the right upper arm and forearm. The quaternion representation of the human is preferred over a representation in angles, such as roll-pitch-yaw or angle/axis, because all elements of a quaternion are bounded between -1 and 1, necessary for inputs to many filtering and prediction methods. When using a filtering method such as a Kalman Filter, it is desirable for the inputs to be continuous. An angle representation normalized between $\pm\pi$ would have discontinuities when angles cross $\pm\pi$. However, quaternions do not suffer from discontinuities. If the angles were not normalized, so the angles could be continuous, then it is possible for the angle to be unbounded if a vector appeared to perform many revolutions. Considering a method

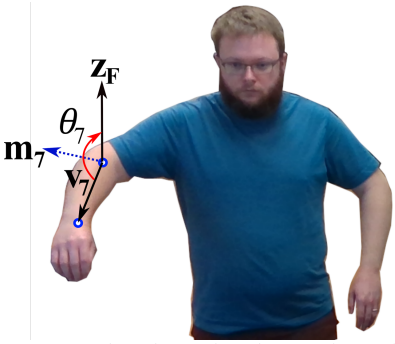

Figure 5. Vectors used to determine the rotation axis and angle for the right forearm.

such as an RNN used to predict joint locations, if the inputs to the RNN are unbounded then it would be impossible to construct a training set of data to cover the entire test data set because the test data set could be infinitely large.

The human's nominal link lengths are taken to be $ll_l = \|\boldsymbol{v_l}\|$, for each link l, for reconstructing the human skeleton from the quaternions and lengths. Therefore, once a human enters a camera's field of view, distances between human joints (link lengths) are determined for each camera frame. From the distances, ratios of link length divided by the spine length are computed for each link. After distance ratios have been computed for at least 20 camera frames, and once the sum of standard deviations of distance ratios over the last 20 frames is below 0.1 and all distance ratios are less than 1, the nominal human link lengths are then taken to be the average distance between joints over the last 20 frames. Requiring the distance ratios to be less than 1 proved to a be a good test for accuracy of the captured link lengths because the spine length is the longest link length for most humans, so all other link lengths should be less than the spine length. Requiring the sum of standard deviations of distance ratios to be less than 0.1 ensures stable distance measurements are captured.

2.4 Fusing Skeletons

After determining the quaternion representation and link lengths of a human, the quaternion representations and link lengths of the same skeleton from multiple viewpoints can be fused to generate one human skeleton. The nominal link lengths of the same skeleton from each camera viewpoint, captured according to the procedure stated earlier, are averaged to generate the fused skeleton link lengths. The quaternions for the fused skeleton are selected from the different camera viewpoints according to the pseudo code in Fig. 6. In this pseudo code, $\boldsymbol{ll_{fused}}$ is the vector of link lengths for the fused human, $\boldsymbol{ll_{camera}}$ is the vector of link lengths for the human determined for each camera, and $\Delta\boldsymbol{ll}$ is the difference between fused link length and the camera link length, for each link. The $\boldsymbol{q_{fused}}(i)$ is the quaternion corresponding to link i in the fused skeleton and $\boldsymbol{q_{cam}}(i)$ is the quaternion corresponding to link i from each camera's data. The $\Delta q = \sum|\boldsymbol{q_{fused}}(i)[k] - \boldsymbol{q_{cam}}(i)[k]|$ is the summation of the differences between quaternion elements (w,x,y,z). For each link of the fused skeleton, the quaternion used

For each camera:
  If time since new camera frame < 0.1sec:
    $\Delta \boldsymbol{ll} = \left|\boldsymbol{ll}_{\boldsymbol{fused}} - \boldsymbol{ll}_{\boldsymbol{camera}}\right|$
    For each link (i):
      If $\Delta \boldsymbol{ll}(i) < 0.5\boldsymbol{ll}_{\boldsymbol{fused}}(i)$ & $\left|v_{\theta_i}\right| < v_{\bar{\theta}}$ & $\left|v_{m_i}\right| < v_{\bar{m}}$:
        $\Delta q = \sum\left|\boldsymbol{q}_{\boldsymbol{fused}}(i)[k] - \boldsymbol{q}_{\boldsymbol{cam}}(i)[k]\right|, k \in \{w, x, y, z\}$
        If $\Delta q < \Delta \boldsymbol{q}_{\boldsymbol{fused}}(\boldsymbol{i})$:
          $\Delta \boldsymbol{q}_{\boldsymbol{fused}}(i) \leftarrow \Delta q, \boldsymbol{q}(i)_{\boldsymbol{fused}} \leftarrow \boldsymbol{q}_{\boldsymbol{cam}}(i)$,

Figure 6. Pseudocode for skeleton fusion.

is the quaternion from any camera that is closest to the previous quaternion of the fused skeleton. For a camera to be considered, it must have provided a new frame in the last 100 milliseconds. Testing indicated OpenPose was the tracking software with the slowest update rate, as low as 15Hz. Therefore, requiring updates at least as fast as every 100 milliseconds is a conservative requirement to allow for skeleton updates as slow as the slowest tracker, with a small buffer, while still ensuring that new skeleton data is being generated by the tracker for each camera. If a camera had stopped reporting new data, possibly due to becoming disconnected or other hardware fault, then the skeleton fusion algorithm would detect that more than 100 milliseconds elapsed without a new skeleton from that camera. Then the fusion algorithm would stop using that camera's data for the fused skeleton, until a new skeleton is reported from that camera.

Another constraint is that for each i^{th} link from each camera, the change in link length, $\Delta \boldsymbol{ll}(i)$ for the i^{th} human link, must be less than half the nominal link length, $\boldsymbol{ll}_{\boldsymbol{fused}}(i)$ for the i^{th} human link. In other words, the observed link length must be within 50% of the nominal link length recorded under ideal conditions. Additionally, the absolute rate of change of rotation angle of Eq. 15 (v_{θ_i}) must be below an upper threshold ($v_{\bar{\theta}}$) and the tip speed of the unit rotation vector of Eq. 12 (v_{m_i}) must be below an upper threshold ($v_{\bar{m}}$) for the quaternion of the i^{th} link from each camera to be considered in generating the fused skeleton, both computed using a derivative of observations. These conditions filter out erroneous data that are clearly not representative of the human skeleton and prevent movements of the fused skeleton that are unrealistically fast. E.g., when a joint of the human becomes occluded, there is a step change in depth resulting in unrealistically high velocity.

2.5 Filtering

After generating the fused set of quaternions, the quaternions of the fused human are passed through a second order low pass Butterworth filter to smooth potentially noisy quaternion elements. The ideal cutoff frequency of 0.25 times the sampling frequency was determined by trial and error. Lower cutoff frequencies caused a visible lag of the fused skeleton compared to the raw skeletons from the trackers. Higher frequencies led to more noticeable jitter in the fused skeleton quaternion elements. Additionally, an $\alpha - \beta - \gamma$ Kalman filter was implemented due to its effectiveness at further filtering out noise, ease of implementation, and option for state extrapolation [19]. The state estimation equations are:

$$\hat{\boldsymbol{x}}_{\boldsymbol{n,n}} = \hat{\boldsymbol{x}}_{\boldsymbol{n,n-1}} + \alpha\left(\boldsymbol{z}_{\boldsymbol{n}} - \hat{\boldsymbol{x}}_{\boldsymbol{n,n-1}}\right) \quad (16)$$

$$\hat{\dot{\boldsymbol{x}}}_{\boldsymbol{n,n}} = (1-\delta)\hat{\dot{\boldsymbol{x}}}_{\boldsymbol{n,n-1}} + \beta\left(\frac{\boldsymbol{z}_{\boldsymbol{n}} - \hat{\boldsymbol{x}}_{\boldsymbol{n,n-1}}}{\Delta t}\right) \quad (17)$$

$$\hat{\ddot{\boldsymbol{x}}}_{\boldsymbol{n,n}} = (1-\delta)\hat{\ddot{\boldsymbol{x}}}_{\boldsymbol{n,n-1}} + \gamma\left(\frac{\boldsymbol{z}_{\boldsymbol{n}} - \hat{\boldsymbol{x}}_{\boldsymbol{n,n-1}}}{0.5\Delta t^2}\right). \quad (18)$$

The $\boldsymbol{z}_{\boldsymbol{n}}$ is the column vector of measurements, which are the elements of the quaternions for the fused skeleton stacked into a column. The $\hat{\boldsymbol{x}}_{\boldsymbol{n,n}}$, $\hat{\dot{\boldsymbol{x}}}_{\boldsymbol{n,n}}$, and $\hat{\ddot{\boldsymbol{x}}}_{\boldsymbol{n,n}}$ are the estimates of the quaternion elements, derivative of elements, and second derivative of the elements. The α is the gain that controls the rate of change of quaternion elements, β controls the rate of change of the derivative of elements, and γ controls the rate of change of the second derivative of elements. The δ term is selected between 0 and 1 and included in Eq. 17 and Eq. 18 to make the estimated velocity and acceleration decay to 0 if $\boldsymbol{z}_{\boldsymbol{n}}$ stopped changing; plus, to allow the possibility of using state extrapolations as input to the filter in future work. I.e., when the best quaternions from all cameras still cannot generate a seemingly accurate skeleton, application of the Kalman filter can yield extrapolated quaternion values. In the event of an inaccurate link quaterion, the elements of $\boldsymbol{z}_{\boldsymbol{n}}$ corresponding to inaccurate sensed quaternions are replaced with the extrapolations of those quaternions. The δ term is important because the longer extrapolations are used in place of sensed data, the less accurate the extrapolations become. Then, it is desirable for the velocity and acceleration extrapolations to decay to 0 to prevent the position extrapolation from going to infinity before accurate sensed data is available again.

Trial and error showed ideal Kalman filter gains α, β, γ, and δ are 0.5, 0.1, 0.001, and 0.3, respectively. Smaller α caused filtered quaternions to converge too slowly to a constant true position. Smaller β and γ caused the filtered quaternions to react too slowly to changes in angular velocity and acceleration. Larger values of $\alpha, \beta,$ or γ reduced the effect of the filter and led to significant jitter in the quaternion elements.

2.6 Forward Kinematics for the Fused Skeleton

After determining the quaternion/link-length representation of the skeleton from each camera, fusing the skeleton quaternion/link-length data, and then filtering quaternion elements, the quaternions and link-lengths of the fused skeleton can be used to generate the cartesian coordinates of each joint of the human skeleton. Using forward kinematics, the skeleton is stitched together by first selecting the location of the pelvis. The skeleton trackers output the hip locations, so the pelvis was taken to be the midpoint of the hips. The skeleton trackers output an estimated accuracy of the estimation of pixel coordinates for each joint. However, these estimated accuracies did not seem to differ in value when a joint was clearly visible or completely occluded. Therefore, an alternate method of determining the accuracy of the estimated hip locations was developed. This method iterated through all the pixels along a line between the hips and lines between the hips and shoulders, determining the change in depth value between each consecutive pair of pixels. The most accurate pelvis location from any camera was taken to be from the camera that generated the lowest maximum absolute

change in pixel depth along those three lines. Note, if an object was between the human and camera along one of those three lines between hips and shoulders, at least one step change in depth between adjacent pixels exists along those lines.

Once the pelvis location is selected, then quaternions and link-lengths of the fused skeleton can be used to generate the rest of the joint locations. In the forward kinematics process, the location of distal joints is determined starting with the location of proximal joints as follows:

$$P_{distal} = P_{proximal} + ll(i)\boldsymbol{v_i} \tag{19}$$

where $P_{proximal}$ is the location of the joint closest to the pelvis in the human kinematic chain for link i, $ll(i)$ is the link length for link i of the fused skeleton, $\boldsymbol{v_i}$ is a unit vector indicating the direction of link i relative to the fixed z-axis, and P_{distal} is the location of the joint farthest from the pelvis for link i. The link vector $\boldsymbol{v_i}$ is determined using quaternion multiplication:

$$\boldsymbol{v_i} = \boldsymbol{q_{fused}(i)} \boldsymbol{q_{z_f}} \boldsymbol{q^{-1}_{fused}(i)} \tag{20}$$

where $\boldsymbol{q_{z_f}}$ is the quaternion representing the world z-axis, $\boldsymbol{q_{z_f}} = \langle 0,0,0,1 \rangle$, $\boldsymbol{q_{fused}(i)}$ was the quaternion determined to rotate the world z-axis to align with link i of the fused skeleton and $\boldsymbol{q^{-1}_{fused}(i)}$ is the inverse of that quaternion. The w_l part of the resulting quaternion would be zero, making the $\sin(0.5\theta_l)$ term in Eq. 11 equal to one, allowing the unit vector $\boldsymbol{v_i}$ to be the x_q, y_q, z_q quaternion elements resulting from this multiplication. The calculations in Eq. 19 and Eq. 20 are performed on each link of the skeleton, starting from the link most proximal to the pelvis, the spine, and ending with the most distal links, the forearms.

3. EXPERIMENTS AND RESULTS

To test the single camera accuracy and simultaneously test the skeleton fusion accuracy, a human touched their wrists to targets placed throughout the robotic testbed. The Comau e.Do 6-DOF robot's end-effector was the target, although any robot or other form of target of known location could have been used [20]. The manipulator not only served as a target to touch, but also as an occlusion for demonstration of skeleton fusion in static and dynamic settings. The left camera was an Intel D455 and the right camera was an Intel D435i [21], both depth cameras. The location of the cameras relative to the robot can be seen in Fig. 7, just to the left and right of the robot. For both cameras, the frame rate set at 30 Hz and resolution was set at 640x480 pixels, selected due to reaching the GPU memory limits when using higher resolution images. Additionally, the skeleton tracking and fusion algorithms were run on a computer with an Intel i9-10900 CPU and Nvidia GeForce RTX 3070 GPU, used for tracker inference. The skeleton fusion algorithm was programmed in Python 3.6 in a ROS Noetic environment. A Python version of OpenPose and the C++ version of Media Pipe was used for skeleton tracking [10,9]. The velocity limits $v_{\bar{\theta}}$ and $v_{\bar{m}}$ were selected to be 20 rad/sec and 10 m/sec, respectively, determined by trial and error to reject erroneous human movement.

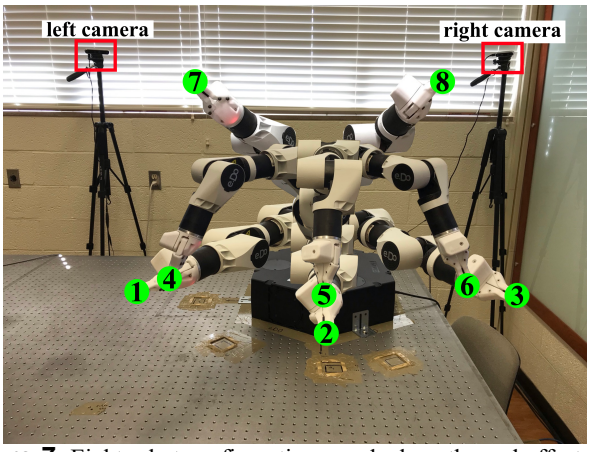

Figure 7. Eight robot configurations used where the end-effector tip was the position target.

Table 2. Robot joint angles and end-effector positions for tests.

Test Position ID	Joint Angles (radians)	End-effector position (x,y,z) (cm)
1	(2.07,1.57,0,0,0,0)	(-37.9,69.6,33.8)
2	(1.57,1.57,0,0,0,0)	(0.1,79.3,33.8)
3	(1.07,1.57,0,0,0,0)	(38.1,69.5,33.8)
4	(2.37,0.6,1.0,0,1.0,0)	(39.4,38.2,23.3)
5	(1.57,0.6,1.0,0,1.0,0)	(0.1,54.9,23.3)
6	(0.77,0.6,1.0,0,1.0,0)	(-39.3,38.3,23.3)
7	(1.17,0.8,0.3,0,0,0)	(26.1,61.7,74.8)
8	(1.97,0.8,0.3,0,0,0)	(-26.1,61.7,74.8)

The 8 test positions are shown as the green circles at the tip of the end-effector in Fig. 7 and are also listed in Table 2. In Fig. 7, the human would be standing in front of the robot such that the human's perspective would be the same as shown in the image. Note that some test positions have similar locations, but more importantly, poses of similar location, such as positions 1 and 4 vary greatly in robot configuration, which is the source of varying levels of occlusion in tests. Stationary touch tests were conducted where the human touched the target at each test position 50 times for each wrist. Dynamic tests were also conducted where the human held their wrist to the robot's end-effector while the robot cycled through the position sequences: 1-3-1 for 20 cycles or 1-4-6-3-1 for 10 cycles for each wrist.

All 8 positions can cause occlusions of the pelvis in one of the cameras' perspectives, depending on where the human is standing relative to the robot. Situations where the pelvis is occluded were necessary for testing since the fused skeleton is built starting with an estimated pelvis position. Test positions 1 through 3 don't cause occlusion of the arms, while positions 4 through 8 can cause occlusion of the arm that touches the target. Two examples are shown in Fig. 8, where images a and b show test position 8, at which the robot occludes part of the torso and left arm from the left camera, but those links are viewable in the right camera's image. Images c and d show position 4 in which the robot occluded part of the torso and left arm in the right camera's image but not in the left camera's image. To create various occlusion situations, when testing each static position,

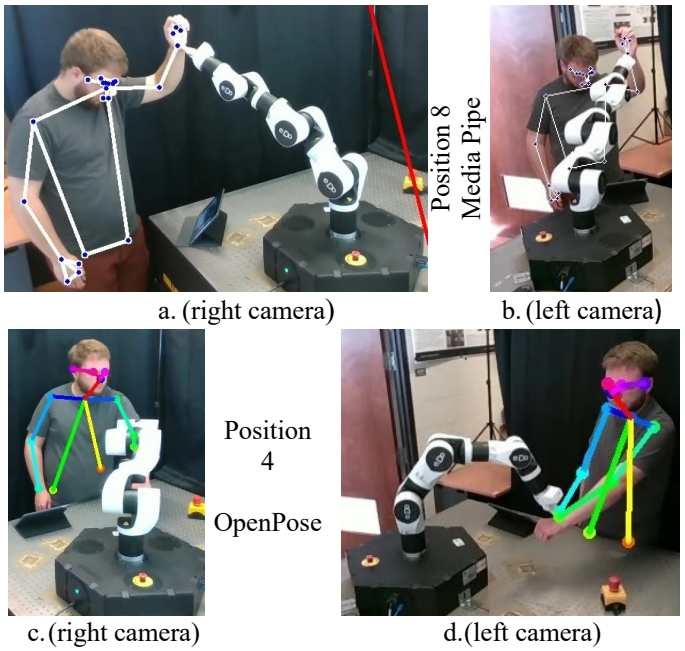

Figure 8. Examples of robot and human positions with which one camera provided inaccurate readings. Images a and b used Media Pipe. Images c and d used OpenPose. Images a and c come from the right camera and b and d come from the left camera.

the human alternated between crossing the test arm across his/her torso and extending the test arm away from the torso.

The results from touching stationary targets indicate that Media Pipe's 3D joint location estimation with the proposed skeleton fusion method generated the lowest position error on average, as shown in Table 3. Additionally, Table 3 shows that the mean and standard deviation of position error is lower for the fused skeleton than for the skeleton determined directly from either the left or right camera's data. Lower standard deviation of position error translates to more repeatable position estimations. Table 4 shows the mean and standard deviation of

Table 3. Average position error between the target and left or right wrist, averaged over all stationary test positions.

Overall Mean / Standard Deviation of Error (cm)			
Tracker	Left Camera	Right Camera	Fused Skeleton
Media Pipe, 2D	21.3 / 14.87	41.2 / 20.40	24.5 / 10.20
Media Pipe 3D	17.0 / 7.38	42.8 / 21.22	**15.5 / 7.21**
OpenPose	32.1 / 42.26	28.5 / 62.07	18.9 / 10.71

Table 4. Average position error between the target and left or right wrist for each stationary test position with the Media Pipe 3D method.

Mean / Standard Deviation of Error (cm)			
Test Pos. ID	Left Camera	Right Camera	Fused Skeleton
1	17.0 / 5.18	38.0 / 13.60	13.9 / 7.57
2	22.5 / 6.03	33.7 / 25.48	17.2 / 7.31
3	20.8 / 4.36	37.8 / 19.31	19.0 / 6.41
4	9.3 / 4.51	67.9 / 8.08	13.5 / 4.80
5	15.8 / 5.41	30.7 / 10.83	14.2 / 6.67
6	15.3 / 8.96	51.9 / 18.44	13.0 / 5.25
7	21.2 / 11.10	48.6 / 9.61	14.9 / 4.40
8	48.5 / 21.51	21.4 / 6.62	18.1 / 10.60

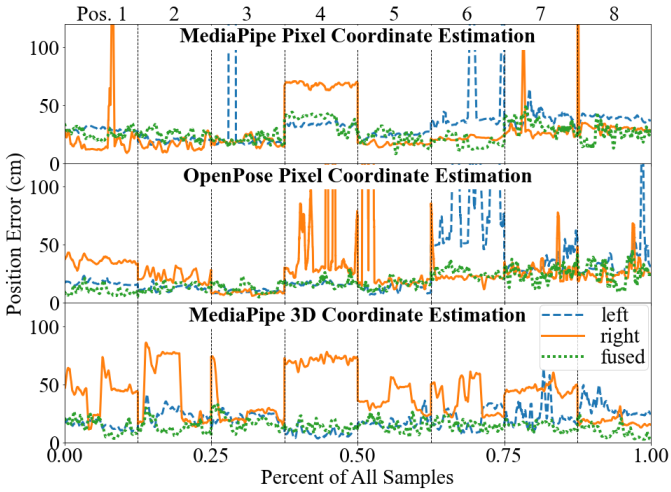

Figure 9. Plots of the position error from all samples for each of the tracking methods.

position error for each of the 8 test positions when using Media Pipe's 3D location estimation, averaged over all 50 samples for right and left wrists. These statistics indicate that for the left camera, test position 8 was least accurate and position 4 had the best accuracy. Interestingly, for the right camera, position 4 was least accurate and position 8 had the best accuracy. Figure 9 shows plots of the position error between the right wrist and the target over all stationary test positions using the skeleton directly from the left camera (dashed blue line), skeleton directly from the right camera (solid orange line), and the fused skeleton (dotted green line). These plots show that for some test positions, such as 4, the left camera provided better position accuracy. And, for other test positions, such as 8, the right camera provided the better position accuracy. The line for the fused skeleton mostly followed the line of the more accurate camera for the entire plot.

Table 5 shows the results from touching the end-effector while the target moved between test positions. These results show that the fused skeleton generated when using OpenPose generated less position error compared to either Media Pipe methods, contrary to Table 3 results. The fusion algorithm with OpenPose generated 16.7cm average position error in dynamic tests, compared with 26.6cm and 28.8cm with Media Pipe joint pixel coordinate estimation and Media Pipe 3D joint coordinate estimation, respectively. Table 5 shows the fused skeleton generated using OpenPose and the fusion algorithm was more accurate than the skeleton directly from either camera. However, when using either Media Pipe method, the fused skeleton was less accurate than the skeleton directly from either camera.

Discussion. Portions of the resulting position errors may be attributed to the following sources. First, the robot position data

Table 5. Average position error between the target and left or right wrist from dynamic position tests.

Overall Mean Error (cm)			
Tracker	Left Camera	Right Camera	Fused
Media Pipe, 2D	22.3	24.0	26.6
Media Pipe 3D	20.4	32.4	28.7
OpenPose	19.6	22.6	**16.7**

relative to the world coordinate frame was not exactly correct. Two robot positioning tests were conducted in which the robot was commanded to go to joint angles that positioned the tip of the end effector at (42.2,0.0,0.0)cm and (0.0,42.2,0.0)cm. The actual measured position for the first test was (42.8,-0.4,0.4)cm and for the second test was (1.8,42.6,0.4)cm, corresponding to a maximum of 1.9cm in Euclidean norm of robot position error. Second, the wrists of the human conducting the tests were about 6.2cm maximum diameter. The depth channel of the cameras only reported the distance between the camera and surfaces nearest the cameras. Consequently, the observed coordinates of the wrist were not at the exact center of the wrist, but instead on the surface of the wrist nearest the camera. A diameter of the wrists, 6.2cm, can be added to the joint position error depending on if the wrist was in front of or behind the target, relative to the camera perspective. Here, the combined error due to both robot position and wrist diameter introduces 8.1cm of error. In the best-case scenario, taking this error into account reduces the minimum average wrist tracking error to 7.4cm, but the error could also increase the tracking error.

In testing it was observed that when a joint was occluded, the trackers often estimated a pixel coordinate for the occluded joint that was just beside the occlusion, resulting in a poor estimation, as seen by the left hip in Fig. 8.c. The results in Table 4 for tracking error at each position with Media Pipe 3D estimation illustrate another potential problem with always using the pelvis as the foundation of the skeleton. They show that position 4 had the least accurate sensing by the right camera because in this position the robot could be between the pelvis and the right camera. This caused depth of the pelvis to be inaccurate. Since the pelvis location was used to relate Media Pipe's estimated 3D joint locations to the world frame and also provided the starting point for skeleton forward kinematics after fusion with all trackers, pelvis position error propagates through the entire skeleton. Individual test position results from the other trackers show the same problem. In future work, skeleton fusion could benefit from using the most accurately sensed part of the skeleton, not just the pelvis, as the start for forward kinematics.

While conducting tests, GPU memory utilization, GPU processor utilization, and average update rate of the skeleton fusion algorithm was recorded, as shown in Table 6. The update rate of the fusion algorithm is limited by how frequently new data is determined by the skeleton trackers. The OpenPose skeleton tracker used over 10 times more GPU memory than either Media Pipe tracking method. Additionally, both Media Pipe tracking methods operated at the camera's frame rate of 30 Hz, indicating Media Pipe processed new camera frames at least as fast as the camera reported new frames. OpenPose processed new frames at an average of 17.8 Hz, 59% the rate of Media Pipe.

Table 6. Skeleton tracker performance comparison.

Tracker	GPU mem. usage (MiB)	GPU use (%)	Avg. data update rate (Hz)
Media Pipe, 2D est.	586	74	30.0
Media Pipe, 3D est.	606	77	30.0
OpenPose	7552	66	17.8

Therefore, Media Pipe is preferred over OpenPose considering only computing resource requirements.

When compared to the results in other works on the subject of skeleton fusion with depth camera inputs, the results of this paper's algorithm show significantly larger position error of the fused skeleton joint locations, as shown in Table 7. A key difference between this work and others is the source of occlusion. The methods in [13,14] were tested using self-occlusion caused by the human's pose. Since the robot was used as the occlusion in experiments in this work, the difference in distance from the camera between the human and robot could be larger than if the human was the source of occlusions. When a human's arm occludes other joints, it is likely that the error in depth is relatively small compared to if the occlusion was caused by a different object. Another difference between the experiments used here and in other works is that only 2 depth cameras were used to test this algorithm compared to 3 and 5 depth cameras used to test other algorithms. Adding more depth cameras placed so that, at all times, at least 1 camera can sense each human link without occlusion could improve results.

Table 7. Comparison of results with competitive methods.

Method [Ref.]	Number Cameras	Occlusion Source	Fused Skeleton Position Error (cm)
[11]	2	Self-occlusion	Did not report
[12]	2	Objects and self	Did not report
[13]	5	Self-occlusion	6.95
[14]	3	Self-occlusion	5.68
Ours	2	Objects (robot)/self	15.3

Based on the results observed from experiments, Media Pipe's 3D joint coordinate estimation provides lower human joint tracking error if the target is stationary. In a manufacturing setting, a stationary target could be the end effector location for a static robot-human handover. However, if the target is moving or tracking moving human joints is desired, then OpenPose skeleton tracking provides lower joint position error than Media Pipe. Many manufacturing settings require tracking of moving human joints. This is necessary for collision detection and avoidance as well as human-robot interaction such as dynamic handovers. However, since the results showed the minimum average tracking error was 16.7cm with a dynamic target and 15.5cm with a static target, skeleton tracking with depth camera inputs may only be appropriate for course collision detection and avoidance. Human-robot interaction requires higher precision to ensure safety of humans and accurate completion of interactive tasks. Therefore, future efforts will investigate other sensing solutions that can provide higher accuracy, but without requiring the worker to wear a large array of sensing hardware.

4. CONCLUSION

In summary, this work presented a novel algorithm for fusing human joint locations generated with skeleton tracking software and multiple depth camera inputs. Media Pipe (2D and 3D joint estimation) and OpenPose skeleton tracking software was tested for providing input to the fusion algorithm. Results showed, with any of the 3 skeleton tracking methods, the fusion

algorithm overall generated wrist locations with less mean position error and standard deviation in error than the wrist location determined by the trackers directly from individual camera data. Results also showed that Media Pipe's 3D joint location estimation generated the lowest mean position error when touching stationary targets. OpenPose generated the lowest mean position error when the wrist followed a moving target. To conclude, the fusion method mitigated the impact of occlusions; but is tracker algorithm dependent for static versus dynamic tracking performance. Future work is still needed to further reduce errors in skeleton tracking algorithms using inexpensive depth cameras. Alternate sensor solutions may still be required for precision human-robot interaction in HRC work cells.

ACKNOWLEDGEMENTS

Funding was provided by the NSF/NRI: INT: COLLAB: Manufacturing USA: Intelligent Human-Robot Collaboration for Smart Factory (Award I.D. #:1830383). Any opinions, findings and conclusions or recommendations expressed are those of the researchers and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Zhang, Jianjing, Liu, Hongyi, Chang, Qing, Wang, Lihui, Gao, Robert X. "Recurrent Neural Network for Motion Trajectory Prediction in Human-Robot Collaborative Assembly." *Cirp Annals-Manufacturing Technology*. Vol. 69 No. 1 (2020): pp. 9–12. DOI 10.1016/j.cirp.2018.04.066

[2] Callens, Thomas, Van der Have, Tuur, Van Rossom, Sam, De Schutter, Joris and Aertbeliën, Erwin. "A Framework for Recognition and Prediction of Human Motions in Human-Robot Collaboration Using Probabilistic Motion Models." *IEEE Rob. Autom. Lett.* Vol. 5 No. 4 (2020): pp. 5151-5158. DOI 10.1109/LRA.2020.3005892.

[3] Schydlo, Paul, Raković, Mirko, Jamone, Lorenzo and Santos-Victor, José. "Anticipation in Human-Robot Cooperation: A Recurrent Neural Network Approach for Multiple Action Sequences Prediction." *Proceedings of the IEEE ICRA*. pp. 5909-5914. Brisbane, Australia. May 21-25, 2018. DOI 10.1109/ICRA.2018.8460924.

[4] Al-Amin, Md., Qin, Ruwen, Moniruzzaman, Md., Yin, Zhaozheng, Tao, Wenjin and Leu, Ming C. "An Individualized System of Skeletal Data-Based CNN Classifiers for Action Recognition in Manufacturing Assembly." *J Intell Manuf* (2021) DOI 10.1007/s10845-021-01815-x.

[5] Flowers, Jared and Wiens, Gloria. "Robot Risk of Passage Among Dynamic Obstacles." *Proceedings of the ASME MSEC*. Virtual, Online, June 21-25, 2021. DOI 10.1115/MSEC2021-63670.

[6] Lavit, Matteo, Ambrosetti, Roberto, Wiens, Gloria J. and Fassi, Irene. "Human–Robot Collaboration in Smart Manufacturing: Robot Reactive Behavior Intelligence." *J Manuf Sci Eng.* Vol. 143 No. 3. (2021). DOI 10.1115/1.4048950.

[7] González, Leticia, Álvarez, Juan C., López, Antonio. M., and Álvarez, Diego. "Metrological Evaluation of Human-Robot Collaborative Environments Based on Optical Motion Capture Systems." *MDPI Sensors.* Vol. 21 No. 11. (2021): pp. 3748-3763. DOI 10.3390/s21113748.

[8] Moniruzzaman, Md., Yin, Zhaozheng, Bin Hossain, Md Sanzid, Guo, Zhishan and Choi, Hwan. "Wearable Motion Capture: Reconstructing and Predicting 3D Human Poses from Wearable Sensors." TechRxviz. Preprint. 2021. DOI 10.36227/techrxiv.16921444.v1.

[9] Bazarevsky, Valentin, Grishchenko, Ivan, Raveendran, Karthik, Zhu, Lixuan Tyler, Zhang, Fangfang and Grundmann, Matthias. 2020. "BlazePose: On-device Real-time Body Pose tracking." ArXiv abs/2006.10204.

[10] Cao, Zhe, Simon, Tomas, Wei, Shih-En and Sheikh, Yaser. "OpenPose: Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields." *IEEE Trans. Pattern Anal. Mach. Intell.* Vol. 48 No 1. (2019): pp.172-186. DOI 10.1109/TPAMI.2019.2929257.

[11] Takahashi, Kosuke, Mikami, Dan, Isogawa, Mariko and Kimata, Hideaki. "Human Pose as Calibration Pattern: 3D Human Pose Estimation with Multiple Unsynchronized and Uncalibrated Cameras." *Proceedings of the IEEE/CVF CVPRW.* pp. 1775–1782. Salt Lake City, UT, June 18-22, 2018. DOI 10.1109/CVPRW.2018.00230.

[12] Chen, Ning, Chang, Yuqing, Liu, Haiqiang, Huang, Lingtao and Zhang, Hongyan. "Human Pose Recognition Based on Skeleton Fusion from Multiple Kinects." *Proceedings of the IEEE CCC.* pp. 5228-5232. Wuhan, China. July 25-27, 2018. DOI 10.23919/ChiCC.2018.8483016.

[13] Moon, Sungphill, Park, Youngbin, Ko, Dong Wook and Suh, Il Hong. "Multiple Kinect Sensor Fusion for Human Skeleton Tracking Using Kalman Filtering." *Int. J. Adv. Robot. Syst.* Vol. 13 No. 2 (2016). DOI 10.5772/62415

[14] Huang, Ching-Chun and Nguyen, Manh-Hung. "Robust 3D Skeleton Tracking based on OpenPose and a Probabilistic Tracking Framework." *Proceedings of the IEEE ICSMC*. pp. 4107-4112. Bari, Italy. Oct. 6-9, 2019 DOI 10.1109/SMC.2019.8913977.

[15] Kálmán, Rudolf E. "A New Approach to Linear Filtering and Prediction Problems" *ASME J. Basic Eng.* Vol. 82 No. 1. (1960): pp. 34-45. DOI 10.1115/1.3662552.

[16] Roux, Joël Le. "An Introduction to Kalman Filter: Probabilistic and Deterministic Approaches." University of Nice, Nice, France (2003).

[17] An, Weizhi, Yu, Shiqi, Makihara, Yasushi, Wu, Xinhui, Xu, Chi, Yu, Yang, Liao, Rijun and Yagi, Yasushi. "Performance Evaluation of Model-Based Gait on Multi-View Very Large Population Database With Pose Sequences." *IEEE Trans. Biom. Behav. Identity Sci.* Vol. 2 No. 4 (2020): pp. 421-430. DOI 10.1109/TBIOM.2020.3008862.

[18] Google (2020) MediaPipe Pose. MediaPipe. https://google.github.io/mediapipe/solutions/pose.html.

[19] Tenne, Dirk, and Singh, Tarunraj. "Characterizing Performance of α-β-γ Filters," *IEEE Trans. Aerosp. Electron. Syst.* Vol. 38 No. 3 (2002): pp. 1072-1087. DOI 10.1109/TAES.2002.1039425.

[20] Comau S.p.A. (October 2017) e.Do Technical Sheet.

[21] Intel RealSense (June 2020) D400 Series Datasheet.

